Methods for operating a CPU having an internal data cache

Field of the invention

The present invention relates to methods for using a CPU cache memory, to chipsets which perform the method, and to devices incorporating those

5   chipsets.

Background of the Invention

Conventionally, a CPU integrated circuit which is a CPU has access to two forms of memory: a first memory ("data cache") within the CPU, and an

10   external memory stored on one or more memory chips. The processor of the CPU is able to access the data cache more quickly than the external memory, but the capacity of the data cache is smaller than that of the external memory, and for many calculations the CPU requires the full capacity of the external memory. For this reason, the data cache is arranged to store a duplicate of

15   the data in the external memory. The processor can then access the data cache when it requires the duplicated data, and the external memory when it requires data which is not duplicated in the internal memory.

The configuration is illustrated in Fig. 1 which shows that when the processor

20   1 of the CPU 3 needs to access data with a given address it does so via a cache controller 5 which determines whether the data with that address is stored in the data cache 7. If so, the data is read from there. If not, the data is read from external memory 9 (which may take the form of one or more memory chips) using a system bus 11. Once the CPU 3 reads data from the

25   external memory 9, the data will be mirrored in the data cache 7, and the next read access from the same location in the external memory 9 will be skipped and instead the data will be taken from the data cache 7. When the processor writes data into the data cache 7 it is not normally copied into the external

memory 9 at once, but instead written into the external memory 9 when the corresponding data in the data cache 7 is going to be replaced. Note that the system of Fig. 1 normally includes an address decoder (not shown) which is involved in passing the signals between the CPU 3 and the external memory

5   9. This address decoder (not shown) is employed because the external memory 9 may be partitioned, and the address decoder defines a mapping between the address in the CPU memory space and an address in the external memory 9. For example, when the CPU 3 issues a write instruction the decoder converts the output address of the CPU 3 in the memory space of

10   the CPU into the address of the corresponding location in the partitioned external memory 9.

The memory address map of the system of Fig. 1 in the memory space of the CPU is illustrated in Fig. 2. All of this memory corresponds to addresses in the

15   external memory 9 (under the mapping defined by the decoder). The address space includes a "memory" section 4 which the processor 1 employs as the read/write memory for performing its operations (varying portions of this memory space are mirrored in the data cache 7), and a "reserved" section 6 which is an area of memory which is not in fact required for performing

20   calculations.

Summary of the invention

The present invention aims to provide a new and useful way of accessing a

25   . memory, and a chip set which implements the method.

The present invention is based on the realisation that the processes of the computer could be implemented with higher performance if only the memory stored in the data cache is used. For example, in a system in which there are several processors, performance would be improved if one or more of those

processors used only their respective internal data caches. This higher performance may result from either or both of faster memory access and reduced external memory requirements.

In general terms, the present invention proposes that a CPU having an internal data cache is operated in combination with a dummy interface which simulates the existence of an external memory having the same address space as the cache memory but which does not store data written to it. In this way, a CPU according to present designs can be operated without the ability to store data in an external memory in respect of at least part of its memory address space.

Specifically, a first expression of the present invention is a data processing system having:

at least one processor chip including a processor unit and an internal data cache, and

a dummy interface which receives data written to it by the processor chip, and discards it.

The processor(s) can also be connected, e.g. via the dummy interface, to an external memory chip which provides data to initialise the data cache.

The at least one processor chip may be one or more of a plurality of processor chips which also includes at least one processor chip having access to an external memory store.

A second expression of the invention is a method of operating a processing chip having a processor, an internal data cache and a cache controller for transmitting write instructions out of the integrated circuit for updating an external memory, the method including discarding the write instructions and

arranging for the program code operated by the processor to require only the data cache as memory.

Brief description of the figures

An embodiment of the invention will now be described for the sake of
5   illustration only with reference to the following figures, in which:

Fig. 1 shows schematically a known data processing system;

Fig. 2 shows the address map for the system of Fig. 1;

Fig. 3 shows schematically a data processing system according to the present
10   invention;

Fig. 4 shows an external memory simulated by the system of Fig. 3;

Fig. 5 shows the address map for the system of Fig. 3; and

Fig. 6 shows a system incorporating an arrangement as shown in Fig. 3.

Detailed Description Of The Embodiments

15   The embodiment is illustrated in Fig. 3, in which components which may be
identical to those shown in Fig. 1 are illustrated by the same reference
numerals. As shown in Fig. 3, the data processing system includes a CPU
processor chip 3 having an internal processor 1, a cache controller 5 and an
internal data cache 7 connected via the system bus 11 to a dummy interface
20   13. The dummy interface 13 is coupled to a memory 19, and to an address
decoder 15.

In an initialisation stage, the dummy interface 13 reads data (including
program code) which is required to initialise the CPU 3 from the memory 19
into part of the data cache 7.

Subsequently, in operation, the processor 1 operates exactly as in present systems. However, the program code which it runs (i.e. the program code uploaded from the memory 19 during the initialisation) is such that the amount of memory required by the processor 1 is provided by the data cache 7. The cache controller 5 operates in the conventional manner, performing write operations to the data cache 7 and attempting to keep an (in fact non-existent) external memory 17 updated to mirror what is in the data cache 7 by transmitting write operations to the system bus 11. These write instructions are passed to the dummy interface 13 which simply discards the data it receives.

The address decoder 15 operates in a way similar to that of the prior art decoder discussed above, but with the difference that it never directs read/write commands from the CPU to the external memory 9. Write commands from the CPU 3 are simply discarded by the dummy interface 13 and the decoder 15. In the case that the dummy interface receives a read command from the CPU 3 (as in some embodiments it may do during the initiation procedure), the dummy interface transmits an output which is not dependent on the data which has previously been transmitted to it, e.g. it may always output "0". This feature may be used during an initialisation of the data cache 7 in which portions of the memory which are not copied from the external memory 19 may be initialised to zero using values read from the dummy interface 13.

We now analyse the situation from the perspective of the CPU 3. The operation of the CPU is identical to what it would be if the dummy interface were not just discarding data, but instead was a normal interface performing read/write operations to a memory 17, operating as shown in Fig. 3: that is, not storing ("N/C") any data written to it, and always outputting the value "0" in

any read operation. In fact, this memory 17 does not exist, but the CPU 3 has no way of "knowing" this.

Since the program code is such that the only memory required is provided by the data cache 7, the processor 1 operates in exactly the same manner as it
5   would in the system of Fig. 1 running the same code. However, the system of Fig. 3 has the advantage that less memory is required, since the external memory 19 of Fig. 3 is only required to store the initialisation data and may be smaller than the memory 9 of Fig. 1.

Turning to Fig. 5, the address map for the system is shown. In this case, the
10   memory space of the CPU 3 is entirely defined by the data cache 7. A portion 14 of the data cache stores data which was read from the external memory 19 during initialisation, and this section is not overwritten during the operation of the processor 1. A second portion 16 of the data cache 7 is the read/write area which the processor 1 uses to perform its calculations. A portion 18 of
15   data cache 7 is "reserved", i.e. unused in the operations of the processor 1.

Note that the CPU 3 may be one of a plurality of processing units which are part of the data processing system, such as the shown in Fig. 6 having a master processing unit 20 and a plurality of slave processing units 21. In this
20   case, one or more of the CPU processing units (e.g. the master processing unit 20 only) may be provided with an external memory 23 which is kept updated to mirror that processing unit's internal data cache by a conventional method such as that described above in relation to Fig. 1. Other of the CPU processing units (e.g. some or all of the slave processing units 21) may be as
25   discussed above in relation to Fig. 3, i.e. not being able to store data an external memory but instead performing processing only using their internal data caches and transmitting their write instructions to dummy interfaces 25 which discard them. The dummy interfaces initialise the slave processing unit 21 using one or more memory chips 29 which do not store data during the

subsequent processing operation. In this way, the overall performance of the system is improved, since fewer read/write memories 23 are required. At the same time, the processing speed of the system is improved since many of the processors (e.g. the slave processing units 21) never need to wait to obtain data from external memories.

In such a system, the only processor chip(s) 20 which are provided with the capacity to read/write data to/from an external memory 23 are the one(s) which require a read/write memory larger than their internal data cache to perform their tasks, and the program code which the full set of processing chips 20, 21 operate is written such that all tasks which require a memory larger than the internal data cache of the CPUs are allotted to those CPU(s) 20 which have access to an external memory.